

Saving File and Variable Attributes to Attach them to a New Variable and File

[Contents](#) [Previous](#) [Next](#)

Goal: To save the files global attributes together with the variable attribute to use them when writing a derived variable to a new file.

You can **download** and run the **python script redecorate.py** or follow the tutorial and type commands at the **cdat** or **python prompt**.

Let's import the modules we will need

```
import cdms, vcs, cdutil, genutil, cdtime, MA, MV, sys, os
```

and open a dataset we will work with

```
file = os.path.join(sys.prefix,'sample_data/tas_ccsr-95a.xml')
a=cdms.open(file)
```

let's import some data

```
data=a('tas')
print data.shape
```

the output is:

(12, 1, 32, 64)

Now lets get the files global attributes and put them into a dictionary called: file_dic

```
# rip out global attributes and put them in a dictionary called: file_dic
#
list_file=a.attributes.keys()
file_dic={}
for i in range(0,len(list_file)):
    file_dic[i]=list_file[i],a.attributes[list_file[i] ]
```

let's see what's inside the list_file:

```
print list_file
```

['calendar', 'id', 'title', 'production', 'institution', 'Conventions', 'cdms_filemap', 'directory', 'model', 'history']

```
print file_dic
```

```
{0: ('calendar', 'noleap'), 1: ('id', 'none'), 2: ('title', 'AMIP 1 Model Output'), 3: ('production', 'CCSR/NIES AGCM (T21 L20) 1995 '), 4: ('institution', 'Tokyo, Japan'), 5: ('Conventions', ''), 6: ('cdms_filemap', '[[[abs_time,tas,bounds_time],[[0,12,--,tas_ccsr-95a_1979.01-1979.12.nc],[12,24,--,tas_ccsr-95a_1980.01-1980.12.nc]]]]'), 7: ('directory', ''), 8: ('model', 'ccsr-95a'), 9: ('history', 'original not LATS-generated have been rewritten in conformance with GDT conventions (described at http://www-pcmdi.gov/drach/GDT_convention.html') }
```

```
tas_ccsr-95a_1979.01-1979.12.nc tas_ccsr-95a_1980.01-1980.12.nc tas_ccsr-95a_1981.01-1981.12.nc  
tas_ccsr-95a_1982.01-1982.12.nc tas_ccsr-95a_1983.01-1983.12.nc tas_ccsr-95a_1984.01-1984.12.nc'}
```

and do the same with the variables attributes

```
#  
# rip out data attributes and put them in a dictionary called: data_dic  
#  
list_data=data.attributes.keys()  
data_dic={}  
for i in range(0,len(list_data)):  
    data_dic[i]=list_data[i],data.attributes[list_data[i]]
```

see the list_data

```
print list_data  
  
['name_in_file', 'missing_value', 'name', 'datatype', 'subgrid', 'long_name', 'units', 'axis']
```

and in the data_dic

```
print data_dic  
  
{0: ('name_in_file', 'tas'), 1: ('missing_value', [ 1.00000002e+20,]), 2: ('name', 'tas'), 3: ('datatype', 'Float'), 4:  
('subgrid', 'time:mean'), 5: ('long_name', 'Surface Air Temperature'), 6: ('units', 'K'), 7: ('axis', 'TZYX')}
```

Now we can derive the annual cycle data

```
#  
# ======  
#  
# calculate Annual Cycle  
#  
cdutil.setTimeBoundsMonthly(data)  
#  
start_time = data.getTime().asComponentTime()[0]  
end_time = data.getTime().asComponentTime()[-1]  
#  
ac=cdutil.ANNUALCYCLE.climatology(data(time=(start_time, end_time, 'cob')))  
#  
# ======
```

And finaly we can put original data attributes back on the calculated annual cycle data

```
#  
# put original data attributes back on calculated annual cycle data  
#  
for i in range(0,len(data_dic)):  
    dm=data_dic[i]  
    setattr(ac,dm[0],dm[1])
```

Now we can write the data to a NetCDF file

```
#  
# write out file and add global attributes to file
```

```

#
o=cdms.open('output.nc', 'w')
o.write(ac)
for i in range(0, len(file_dic)):
    dm=file_dic[i]
    setattr(o, dm[0], dm[1])

o.close()
a.close()

```

Here is a dump of the header information from the file 'outpu.nc', we just wrote:

```

netcdf output {
dimensions:
time = UNLIMITED ;// (12 currently)
bound = 2 ;
zh = 1 ;
latitude = 32 ;
longitude = 64 ;

variables:
double time(time) ;
time:bounds = "bounds_time" ;
time:units = "days since 0" ;
time:calendar = "noleap" ;
time:axis = "T" ;
double bounds_time(time, bound) ;
double zh(zh) ;
zh:positive = "up" ;
zh:long_name = "Height above Earth's Surface" ;
zh:units = "m" ;
zh:axis = "Z" ;
double latitude(latitude) ;
latitude:bounds = "bounds_latitude" ;
latitude:long_name = "Latitude" ;
latitude:units = "degrees_north" ;
latitude:axis = "Y" ;
double bounds_latitude(latitude, bound) ;
double longitude(longitude) ;
longitude:bounds = "bounds_longitude" ;
longitude:topology = "circular" ;
longitude:long_name = "Longitude" ;
longitude:units = "degrees_east" ;
longitude:modulo = 360. ;
longitude:axis = "X" ;
double bounds_longitude(longitude, bound) ;
double tas(time, zh, latitude, longitude) ;
tas:missing_value = 1.000000020040877e+20 ;
tas:name = "tas" ;
tas:subgrid = "time:mean" ;
tas:long_name = "Surface Air Temperature" ;
tas:units = "K" ;
tas:axis = "TZYX" ;

```

```
// global attributes:  
:Conventions = "" ;  
:calendar = "noleap" ;  
:title = "AMIP 1 Model Output" ;  
:production = "CCSR/NIES AGCM (T21 L20) 1995 " ;  
:institution = "Tokyo, Japan" ;  
:cdms_filemap =  
"[[[abs_time,tas,bounds_time],[[0,12,--,tas_ccsr-95a_1979.01-1979.12.nc],[12,24,--,tas_ccsr-95a_1980.01-1980.  
;  
:directory = "" ;  
:model = "ccsr-95a" ;  
:history = "original not LATS-generated have been rewritten in conformance with GDT conventions  
(described at http://www-pcmdi.gov/drach/GDT\_convention.html) [2006-5-5 21:54:46]  
/home/mccoy20/utils/cdat/bin/cdscan -x tas_ccsr.xml tas_ccsr-95a_1979.01-1979.12.nc  
tas_ccsr-95a_1980.01-1980.12.nc tas_ccsr-95a_1981.01-1981.12.nc tas_ccsr-95a_1982.01-1982.12.nc  
tas_ccsr-95a_1983.01-1983.12.nc tas_ccsr-95a_1984.01-1984.12.nc" ;  
}
```

This tutorial was provided by [Jay Hnilo](#)

[Contents](#) [Previous](#) [Next](#)